Team 04: Trainingstracker

Egorow Timothy
Schifferl Kilian
Al Ghazzawi Badee
Friedrich Jeremias
Evangelou Katerina

Datum	Version	Änderung	Autor
02.04.22	1.0	Erstellung	Egorow
02.07.22	2.0	Fehler	Egorow

2. Einführung

2.1 Referenzen

- Vorlesungsmaterialien Software-Engineering
- Craig Larman UML 2 und Patterns

2.2 Übersicht

In diesem Dokument geht es um die Architektur unserer Software. Es werden Ziele und Einschränkungen genannt welche damit verbunden sind. Zuletzt gehen wir auf die Leistungen unserer Software ein.

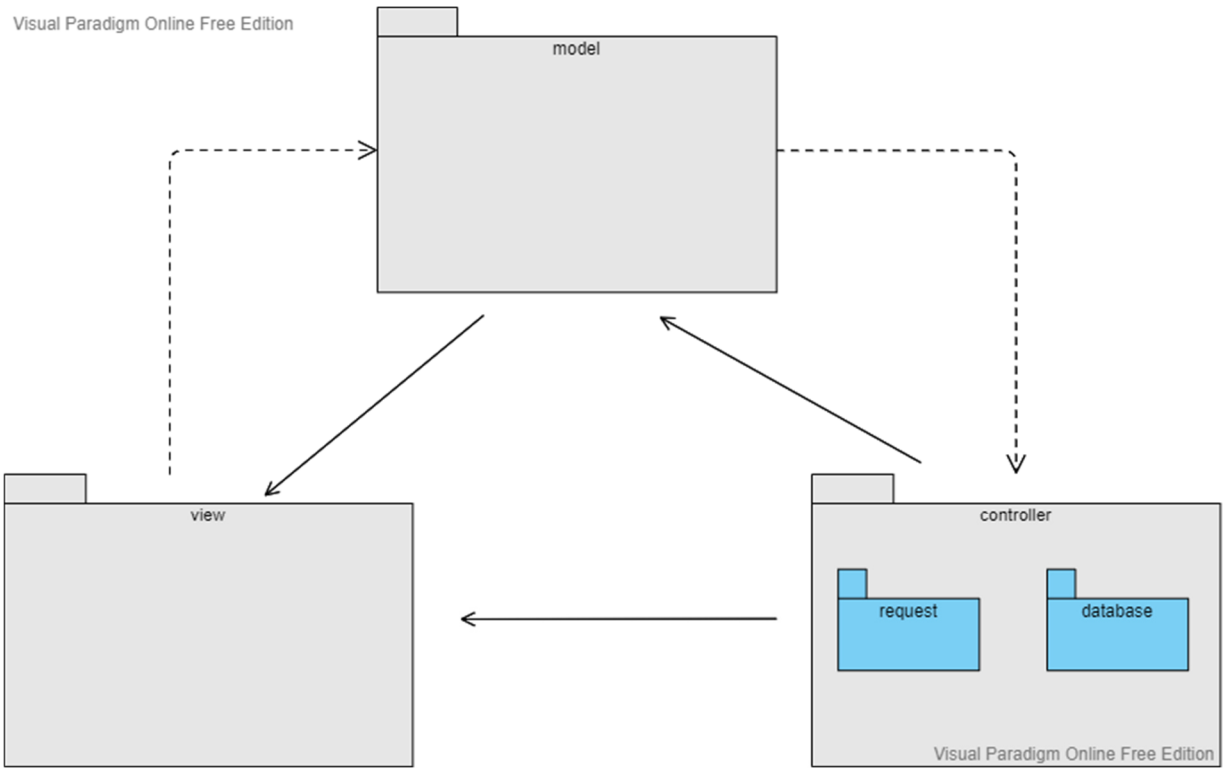
3. Architektonische Darstellung

Für unsere Software haben wir uns für den Model-View-Controller (MVC) entschieden. Dieses Muster teilt die Software in ein Model, darin sind alle Daten enthalten, ein View, dieses dient der visuellen Darstellung, und dem Controller, welcher die Schnittstelle zwischen dem Benutzer und dem System ist, ein.

4. Architektonische Ziele und Einschränkungen

Wir haben uns für den MVC entschieden, da dieser für unser Projekt am besten geeignet ist. Der Grund dafür ist, dass die Komponenten voneinander abhängig sind und dadurch können Änderungen leichter durchgeführt werden.

5. Logische Architektur



In diesem Diagramm sind die Abhängigkeiten erkennbar. Der Controller steuert die Ereignisse im View - Paket und gibt diese an das Model weiter. Dieser teilt sich in 2 Pakete, dem Request und Database auf und steuern Communication zwischen Model und View. Im Model sind alle Daten enthalten. Dieses hat keinen Zugriff auf andere Komponenten. Das View erhält über die Controller, die Daten darstellen zu können. In unserem Projekt schickt der Client eine http Anfrage an den Controller. Dieser wertet diesen aus und führt, wenn alles passt, die Änderungen am Model durch. Sobald die Änderungen durchgeführt wurden, schickt er eine http Antwort an den Client zurück.

5.2 Design Pakete

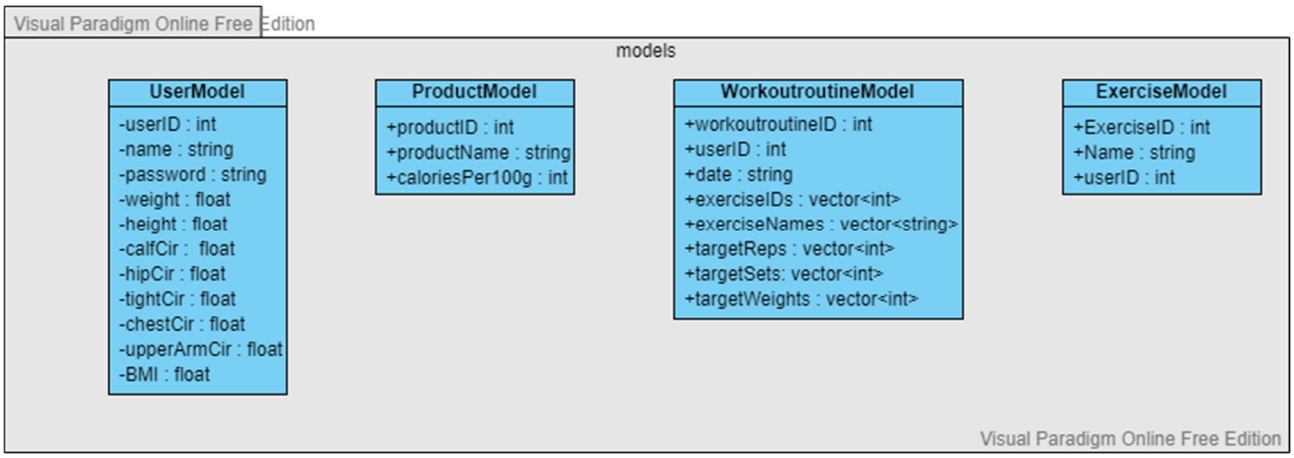
Im folgendem werden die Pakete weiter ausgeführt und erläutert.

5.2.1 View Paket

Login.ui	Fenster zum anmelden
Home.ui	Hauptfenster
Userdaten.ui	Fenster mit Userkörperangaben
Benutzer.ui	Fenster mit Userdaten
Trainingsplan.ui	Fenster zum Erstellen des Trainingsplans
Übung.ui	Fenster mit Übungen
Kalorienzähler.ui	Fenster des Kalorienzählers

Das Paket View ist das Userinterface auf dem Client. Es dient als Schnittstelle zwischen Nutzer und System. Die vom Nutzer eingegebenen Daten werden an den Controller geschickt und dort weiter verarbeitet.

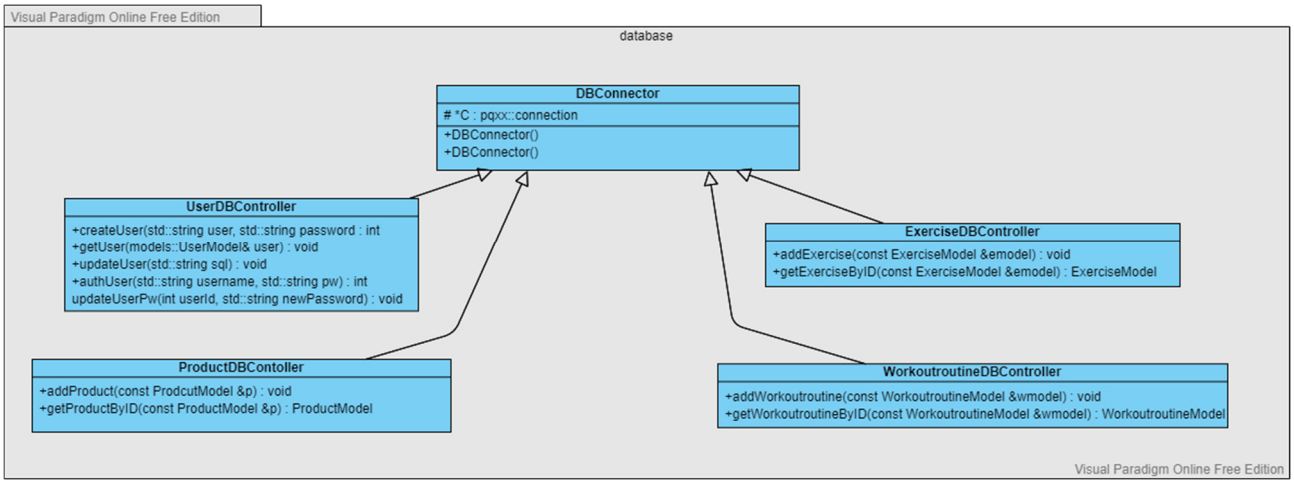
5.2.2 Model Paket

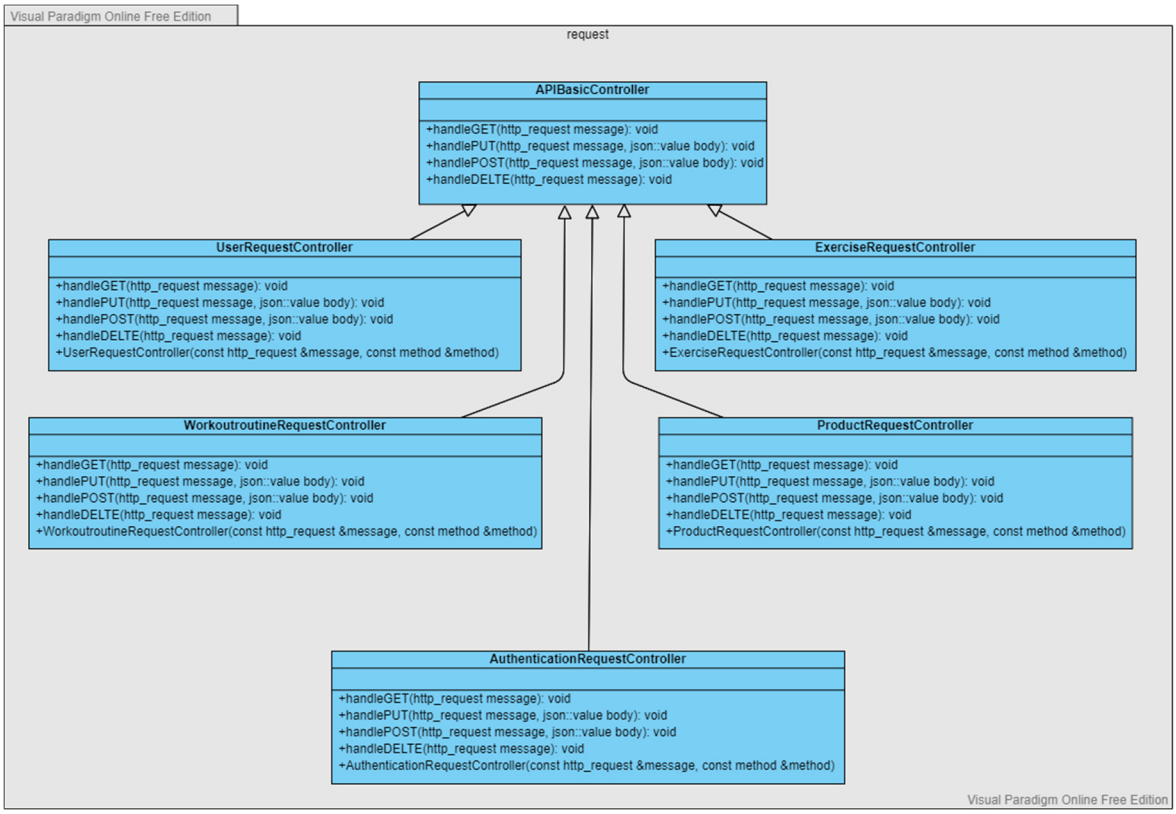


Unser Model-Paket ist dafür da, um Daten in die Datenbank zu schreiben und Daten aus der Datenbank zurück zu erhalten.

UserModel	Enthält Userdaten
WorkoutroutineModel	Enthält Trainingsplandaten
ExerciseModel	Enthält Uebungsdaten
ProductModel	Enthält Produktdaten

5.2.3 Paket Controller





Wir nutzen 2 Controller Pakete, eines für die Datenbank und den API-Server, und eines für den Client und den API Server.

Der Datenbank-Controller ist dazu da, um zwischen dem API-Server und der Datenbank zu kommunizieren. Anfragen vom Client werden mit dem Request-Controller verarbeitet, und dann wird die Anfrage mit Hilfe des Datenbank-Controllers erledigt.

DBConnector	Stellt Verbindung zur Datenbank her
UserDBController	Übergibt Userdaten an die Datenbank
WorkoutroutineDBController	Übergibt Trainingsplan Daten an die Datenbank
ExerciseDBController	Übergibt Übungsdaten an die Datenbank
ProductDBController	Übergibt Produktdaten an die Datenbank

APIBasicController	Schablone der RequestController
UserRequestController	Wandelt Userobjects um
WorkoutroutineRequestController	Wandelt Workoutroutineobjects um
ExerciseRequestController	Wandelt Exerciseobjects um
ProductRequestController	Wandelt Productobjects um
AuthenthicationRequestController	Validiert den User

6. Physikalische Sicht

Aufgrund unseres API-Servers müssen wir die Physikalische Sicht aufteilen. Der Client hat nur eine UI, mit welcher er in der Lage ist, Requests an unseren Server zu senden. Auf unserem Server befinden sich die beiden Controller die die Kommunikation zwischen User-Server-Datenbank ermöglichen. Jegliche Logik befindet sich ebenfalls auf dem Server.

7. Prozesse und Threads

Unsere Software läuft nur auf einem Thread.

8. Datenspeicherung

Unsere Daten werden auf einer Datenbank gespeichert, welche unter PostgreSQL läuft. Der Server baut sich eine Verbindung zu Datenbank mithilfe des DB-Controllers.

Folgende Tables werden genutzt:

Workoutroutine:
workoutroutineID, date, exerciseID, userID

Exercise:
exerciseID, name, userID

Workoutroutine_Exercise:
workoutroutineID, exerciseID, targetReps, targetSets, targetWeight

Users:
userID, name, password, age, weight, calfCir, hipCir, tightCir, chestCir, upperArmCir, bmi

9. Größen und Leistungen

Die Menge an gespeicherten Daten soll die Performance nicht beeinflussen. Um keine Redundanz zu erzeugen, wird überprüft ob eine ID bereits vergeben ist